

This page Is Inserted by IFW Operations  
And is not part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of  
The original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

## **IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
Please do not report the images to the  
Image Problem Mailbox.**

**THIS PAGE BLANK (USPTO)**

09/786388  
JC02 Rec'd PCT/PTO 02 MAR 2001

**CERTIFICATE OF MAILING BY EXPRESS MAIL**

**Express Mail Mailing Label Number EL655301165US**

**Date of Deposit: March 2, 2001**

I hereby certify that this correspondence is being deposited with the United States Postal "Express Mail Post Office to Addressee" service under 37 CFR 1.10(c) on the date indicated above and is addressed to:

**BOX PCT  
Assistant Commissioner for Patents  
Washington DC 20231**

**Case Number: P01,0041  
Applicant(s): Erwin Thurner**

**International Application No. PCT/DE99/02753  
International Filing Date 01 SEPTEMBER 1999  
Priority Date Claimed 02 SEPTEMBER 1998**

**Title: Method for Determining a Graphic Structure of a Technical System and Arrangement and Set of Arrangements for Determining a Graphic Structure**

**Enclosed are the following documents:**

International application as filed, drawings attached;  
English Translation, translated drawings attached;  
Annexes;

PTO 1390 in duplicate;  
Executed Declaration;

Amendment "A" prior to action and Appendix "A";  
Information Disclosure Statement, PTO 1449, Search Report, 04 References;  
Submission of Drawings, 5 sheets of drawings, Figures 1-5;

Substitute Specification and Substitute Specification mark-up;

Fee: \$860.00;  
Postcard.

  
Signature of person mailing documents and fees

**THIS PAGE BLANK (USPTO)**

09/786388

PCT/DE 99/02753

BUNDESREPUBLIK DEUTSCHLAND

**PRIORITY DOCUMENT**  
SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH  
RULE 17.1(a) OR (b)



REC'D 26 NOV 1999

WIPO

PCT

EU

DE 99/2753 **Bescheinigung**

Die Siemens Aktiengesellschaft in München/Deutschland hat eine Patentanmeldung unter der Bezeichnung

„Verfahren zur Bestimmung einer graphischen Struktur eines technischen Systems und Anordnung sowie Satz von Anordnungen zur Bestimmung einer Graphen-Struktur“

am 2. September 1998 beim Deutschen Patent- und Markenamt eingereicht.

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

Die Anmeldung hat im Deutschen Patent- und Markenamt vorläufig das Symbol G 06 F 17/50 der Internationalen Patentklassifikation erhalten.

München, den 20. Oktober 1999

Der Präsident des Deutschen Patent- und Markenamts

Im Auftrag

Nietiedt

Aktenzeichen: 198 39 972.3

## Beschreibung

Verfahren zur Bestimmung einer graphischen Struktur eines  
technischen Systems und Anordnung sowie Satz von Anordnungen  
5 zur Bestimmung einer Graphen-Struktur

Es ist bekannt, verschiedene technische Systeme mittels einer graphischen Struktur zu beschreiben.

- 10 Aus [1] ist für ein solches technisches System, einer elektronischen Schaltung, bekannt, die elektrische Schaltung in Form einer graphischen Struktur mit Elementen, die eine elektronische Schaltung beschreiben, zu bestimmen.
- 15 Elemente einer Graphen-Struktur im Rahmen einer Schaltungssimulation sind Symbole, die elektronische Bauelemente symbolisieren, beispielsweise einen Widerstand, einen Kondensator, eine Induktivität, einen Transistor, einen Operationsverstärker oder andere, aus diesen Elementen zusammengesetzte elektronische Bauelemente.
- 20

Bei dem aus [1] bekannten Verfahren und der aus [1] bekannten Anordnung werden von einem Editor-Programm einem Benutzer zur Verfügung gestellte Elemente zur graphischen Beschreibung einer elektronischen Schaltung ausgewählt derart, daß mit den ausgewählten Elementen das technische System "elektronische Schaltung" beschrieben wird. Die Elemente werden von dem Editor-Programm dargestellt.

- 30 Eine Graphen-Struktur beschreibt einen Graph  $G (= V, E, \Psi)$ , der eine endliche, nicht leere Menge  $V$  ( $v \in V$  bezeichnen Knoten des Graphen  $G$ ) aufweist sowie eine endliche Menge  $E$  ( $e \in E$  bezeichnen Kanten des Graphen  $G$ ). Die Knoten und Kanten des Graphen werden verknüpft durch eine Inzidenzfunktion  $\Psi$ ,  
35 die gemäß folgender Vorschrift gebildet wird:

$$\Psi: E \rightarrow \{\{i, j\} | i, j \in V\} \quad (1)$$

3

- (i)  $S = \{ s_1, s_2, \dots, s_n \}$  Menge von Stellen
- (ii)  $T = \{ t_1, t_2, \dots, t_m \}$  Menge von Transitionen
- (iii)  $S \cap T = \emptyset$  S und T disjunkt (die Knotenmenge besteht aus S und T)
- (iv)  $F \subseteq (S \times T) \cup (T \times S)$  Flußrelation

Nachteilig an den bekannten Verfahren und Anordnungen ist insbesondere, daß jeweils anwendungsabhängig nur für eine spezielle Anwendung vorgesehene Elemente eines Graphen zur Bestimmung der graphischen Struktur eines technischen Systems zur Verfügung gestellt werden. So kann mit dem Editor-Programm aus [1] lediglich eine Auswahl unter Elementen zur Beschreibung einer elektronischen Schaltung und bei dem Editor-Programm aus [2] lediglich eine Auswahl aus Elementen zur Beschreibung eines Petri-Netzes erfolgen.

Ein solches bekanntes Editor-Programm ist somit äußerst unflexibel für den Fall, daß ein Anwender unterschiedliche Arten einer graphischen Struktur zur Beschreibung eines technischen Systems einsetzen möchte. Für jede spezielle Anwendung muß dann ein eigenes, für die Anwendung angepaßtes Editor-Programm entwickelt werden, was zu erheblichen Entwicklungskosten führt.

Somit liegt der Erfindung das Problem zugrunde, ein Verfahren zur Bestimmung einer graphischen Struktur eines technischen Systems sowie eine Anordnung und einen Satz mehrerer Anordnungen zur Bestimmung einer Graphen-Struktur anzugeben, welche gegenüber den bekannten Verfahren und Anordnungen eine verbesserte Flexibilität aufweist.

Das Problem wird durch das Verfahren, die Anordnung sowie den Satz von Anordnungen gemäß den Merkmalen der unabhängigen Patentansprüche gelöst.

- a) eine erste Anordnung, die einen Speicher aufweist, in dem eine Menge mehrerer unterschiedlicher Graphen-Struktur-Dateien gespeichert sind, wobei in einer Graphen-Struktur-Datei jeweils angegeben ist, welche Elemente zu deren Darstellung ausgewählt werden können, um einen Graphen zu bilden, und
- b) eine mit der ersten Anordnung gekoppelte zweite Anordnung, die folgende Komponenten aufweist:
- eine Auswahleinheit, mit der eine Graphen-Struktur-Datei aus der Menge der Graphen-Struktur-Dateien ausgewählt werden kann,
  - ein Editor-Programm, mit dem unter Verwendung aus der Menge der Graphen-Struktur-Dateien ausgewählten Graphen-Struktur-Datei ein Graph mit Elementen der ausgewählten Graphen-Struktur-Datei bestimmt werden kann, womit die Graphen-Struktur bestimmt ist,
  - eine mit dem Editor-Programm gekoppelte Darstellungskomponente, mit der die bestimmte Graphen-Struktur dargestellt werden kann.

Durch die Erfindung wird ein gegenüber den bekannten Verfahren und Anordnungen sehr flexibles Verfahren und eine sehr flexible Anordnung zur Bestimmung einer graphischen Struktur angegeben, welche schnell und unkompliziert an neue Anwendungsszenarien bzw. an bestehende Anwendungsszenarien besser angepaßt werden kann.

Auf diese Weise werden verschiedene Arten von Strukturen, die als Graph darstellbar sind, mit einem Verfahren bzw. mit einer Anordnung auf flexible, kostengünstige und einfache Weise bearbeitbar.

Bevorzugte Weiterbildungen der Erfindung ergeben sich aus den abhängigen Ansprüchen.

Das technische System ist vorzugsweise eine elektronische Schaltung oder eine technische Anlage.



Figur 4 ein Ablaufdiagramm, in dem die Verfahrensschritte des Verfahrens gemäß einem Ausführungsbeispiel dargestellt sind;

- 5    Figur 5 ein Satz mehrerer Anordnungen, die gemäß einem zweiten Ausführungsbeispiel über ein Kommunikationsnetz miteinander gekoppelt sind.

10    Fig.1 zeigt eine Anordnung 100 mit einer Menge 101 mehrerer unterschiedlicher Graphik-Struktur-Dateien 102, 103, 104, 105. Jede Graphik-Struktur-Datei 102, 103, 104, 105 ist als dynamisch bindbare Datei (dynamic link library) ausgestaltet.

15    Von einem Benutzer 106 wird über eine mit einem Editor-Programm 107 verbundenen Auswahlkomponente 108 (Tastatur und/oder Computermouse) eine Graphik-Struktur-Datei 102, 103, 104, 105 ausgewählt.

20    Die ausgewählte Graphik-Struktur-Datei, in diesem Ausführungsbeispiel eine erste Graphik-Struktur-Datei 103, wird dynamisch in das Editor-Programm 107 eingebunden.

25    Nach Einbindung in das Editor-Programm 107 ist über eine mit dem Editor-Programm 107 verbundene Darstellungskomponente 109 dem Benutzer 106 auf einem Bildschirm 110 eine Menge 111 von auswählbaren Elementen 112, 113, 114, die in der ersten Graphik-Struktur-Datei 103 angegeben sind, als auswählbare Elemente zur Bestimmung eines weiteren beschriebenen Graphen dargestellt. Ferner sind in diesem Ausführungsbeispiel gemäß  
30    der ersten Graphik-Struktur-Datei 103 ein erstes Überprüfungsprogramm 115 sowie ein zweites Überprüfungsprogramm 116 in dem Editor-Programm 107 eingebunden und werden dem Benutzer 106 zur Auswahl zur Verfügung gestellt.

35    Jede Graphik-Struktur-Datei 102, 103, 104, 105 weist jeweils eine Menge auswählbarer Elemente für die jeweilige Art von Graphen auf, wobei jeweils eine Graphik-Struktur-Datei vorge-

Über eine Menge 206 weiterer, im weiteren näher erläuteter  
Auswahlelemente ist die Auswahl und Bearbeitung spezifischer  
Elemente für ein Petri-Netz 201 dem Benutzer 106 zur Verfü-  
5 gung gestellt.

Ein zweites Auswahlelement 207 ist beschrieben durch ein lee-  
res Rechteck und symbolisiert eine zeitbehaftete Transition.

10 Ein drittes Auswahlelement 208 symbolisiert eine zeitlose  
Transition, die als ausgewählte Transitions-Elemente 220,  
221 und 222 in dem Petri-Netz 201 dargestellt sind.

Ein viertes Auswahlelement 209 symbolisiert eine Kante, die  
15 in diesem Ausführungsbeispiel eine gerichtete Kante ist.

Ein fünftes Auswahlelement 210 symbolisiert eine gemäß der  
Strukturregeln eines Petri-Netzes 201 bezeichnete verbotene  
Kante.

20

Ein sechstes Auswahlelement 211 symbolisiert eine Stelle, wo-  
bei jeweils ein Stellen-Element 223, 224, 225, 226 in dem  
Petri-Netz 201 dargestellt sind. Die Stellen-Elemente 223,  
224, 225 und 226 sind mit den Transition-Elementen 220, 221,  
222 über Kanten 227, 228, 229, 230, 231 und 232 verbunden.

Ein siebtes Auswahlelement 212 symbolisiert die Möglichkeit,  
eine Kombination mehrerer Elemente des Petri-Netzes zu einem  
Gesamtelement zusammenzufassen.

30

Ein achttes Auswahlelement 213 symbolisiert einen Eingang des  
Petri-Netzes 201 und ein neuntes Auswahlelement 214 symboli-  
siert einen Ausgang eines Petri-Netzes 201.

35 Den Kanten sowie den einzelnen Knoten, das heißt den Elemen-  
ten des Petri-Netzes 201 sind textuelle Informationen 251,  
252, 253, 254, 255, 256, 257, 258, 259, 260 und 261 zugeord-

auf, die jeweils über Kanten 317 miteinander verbunden sind. Ferner ist ein Masseanschluß 318 in Fig.3 dargestellt. Den einzelnen Schaltungselementen ist textuelle Information 319, 320, 321, 322, 323, 324, 325, 326 zugeordnet zur näheren Erläuterung der elektronischen Schaltung 310.

Fig.4 zeigt zur Verdeutlichung des Verfahrens das Verfahren in seinen Verfahrensschritten.

10 In einem ersten Schritt (Schritt 401) wird eine Graphik-Struktur-Datei 102, 103, 104, 105 aus einer Menge 101 von Graphik-Struktur-Dateien 102, 103, 104, 105 ausgewählt.

15 In einem zweiten Schritt (Schritt 402) erfolgt eine Auswahl von Elementen, die gemäß der Graphik-Struktur-Datei 102, 103, 104, 105, die in dem Schritt zuvor (Schritt 401) ausgewählt wurde, zur Verfügung stehen.

20 Die ausgewählten Elemente werden von dem Editor-Programm 107 in einem weiteren Schritt (Schritt 403) dargestellt.

Fig.5 zeigt einen ersten Rechner 500 mit einem Speicher 502 und einem Prozessor 503, die jeweils über einen Bus 504 miteinander und mit einer Eingangs-/Ausgangsschnittstelle 501 verbunden sind.

Über die Eingangs-/Ausgangsschnittstelle 501 ist der erste Rechner 500 mit einem Bildschirm 505, einer Tastatur 506 sowie einer Computermaus 507 verbunden.

30 Ferner ist der erste Rechner 500 über ein Kommunikationsnetz 560, in dem Ausführungsbeispiel ein ISDN-Netz (Integrated Services Digital Network) mit weiteren Rechnern 510, 520, 530, 540 und 550 verbunden.

35 In dem ersten Rechner 500 ist die Menge 101 der Graphik-Struktur-Dateien 102, 103, 104, 105 gespeichert.

diglich von der jeweiligen Art des zu bestimmenden Graphen abhängig.

5 Das technische System kann beispielsweise auch eine technische Anlage sein, die durch den Graphen in ihrem Verhalten oder in ihrer Struktur beschreibbar ist.

10 Das Editor-Programm sowie der mit dem Editor-Programm dargestellte Graph kann im Rahmen einer Simulation des technischen Systems eingesetzt werden.

Im weiteren ist eine Realisierung des oben beschriebenen Ausführungsbeispiels angegeben, geschrieben in der Programmiersprache C, wobei die Realisierung in drei Dateien aufgegliedert ist:

5

# 1. Initialisierungs-Datei:

```

package interfaces;

10 import java.io.*;
import java.util.*;
import java.awt.*;

import etc.*;
15 import elements.*;
import mmi.*;
import tools.*;

public class Initialisierung {
20   GraphEditor editor;
    // Der hat die Tokens aus der
    Datei
    StreamTokenizer token;
    // Hier kommen alle erlaubten
25 Knoten und Kanten aus der
    // .lgc Datei rein.
    // Die Einträge werden mit den
    Namen der Objekte referenziert
    Hashtable gobjekte;
30   // Die aktuelle .lgc Datei
    //String configFile;
    // steht jetzt bei den Einstel-
    lungen
    /**
35   * Hier stehen alle Attribute
    drin.
    */
    Hashtable attributNamen;
    /**
40   * hier kommen die Einträge für
    das Menue Tools
    * hinein.
    */
    Hashtable tools;

45   public Initialisie-
    rung(GraphEditor editor) {
        this.editor = editor;
        gobjekte = new Hashtable();
50   attributNamen = new Has-
    htable();
        tools = new Hashtable();
    }

55   /**
    * Diese Methode würde die er-
    ste Initialisierungsdatei
    * einlesen für die Einstellu-
60   gen der Farben, Schriften...
    * Aber ich darf leider nicht.
    */
    /**
    public void readFirst(String
65   name) {
        String configFile = new
        String(name);
        int c;
        //Properties properties = new
70   Properties();
        //properties = Sy-
        stem.getProperties();
        //filename = new String("..")
        + proper-
75   ties.getProperty("file.separator"
        ) + configFile);
        try {
            File file = new
            File(configFile);
80   //FileInputStream in = new
            FileInputStream(file);
            FileReader in = new File-
            Reader(file);
            token = new StreamToken-
85   zer(in);

            //Einstellen der Optionen
            für token
            to-
            ken.eolIsSignificant(true);
            token.quoteChar('"');
            //token.quoteChar('\\');
            //token.quoteChar('(');
            token.quoteChar(')');
95   //Überlese { und , und ;
            to-
            ken.whitespaceChars('(', '(');
            to-
            ken.whitespaceChars(',', ',');
100   to-
            ken.whitespaceChars(';', ';');

            boolean fertig = false;
            while (!fertig) {
                switch
                (c=token.nextToken()){

```

```

17
    if
    (token.sval.equals("SHORTCUTS"))
    {
        while (c != ' ')
5 {
    c=token.nextToken();
        if (c == '"')
10 {
        String
mpunkt = token.sval;

    //System.out.print("MENUPUNKT " +
    token.sval);
15 c=token.nextToken();
        String icon1
    = token.sval;
20 //System.out.print("ICON1 " + to-
    ken.sval);
    c=token.nextToken();
        String icon2
25 = token.sval;

    //System.out.println("ICON2 " +
    token.sval);
    edi-
30 tor.getShortcutableiste().addShortB
    utton();
        }
    }
    break;
35 }
    if
    (token.sval.equals("ACCELERATOR"))
    ) {
        while (c != ' ')
40 {
    c=token.nextToken();
        if (c == '"') {
105 String la-
45 bel = token.sval;

    //System.out.print("MENUPUNKT " +
    token.sval);
110 c=token.nextToken();
        if (c ==
StreamTokenizer.TT_WORD) {
        char cut =
115 token.sval.charAt(0);
55 //System.out.println(" TASTEN " +
    cut);
    edi-
120 tor.getMenuleiste().addShortcutT
60 oVector(label, cut);
        }
    }
}

    }
    break;
65 }
    if
    (token.sval.equals("WINDOWSIZE"))
    {
    c=token.nextToken();
        int x
    =(int)token.nval;
70 c=token.nextToken();
    c=token.nextToken();
        int y
    =(int)token.nval;
75 //size.setSize(x,y);
    break;
80 }
    if
    (token.sval.equals("WINDOWPOSITIO
85 N")) {
    c=token.nextToken();
        int x
    =(int)token.nval;
90 c=token.nextToken();
    c=token.nextToken();
        int y
    =(int)token.nval;
95 //location.setSize(x,y);
    break;
100 }
    if
    (token.sval.equals("AUTHOR")) {
    c=token.nextToken();
        if (c == '"') {
105 Sy-
stem.out.println("AUTHOR " + to-
ken.sval);
        }
    break;
110 }
    if
    (token.sval.equals("TOOLS")) {
        while (c != ' ')
115 {
    c=token.nextToken();
        if (c == '"')
120 {
        String pfa
d=new String(token.sval);
    //System.out.println("TOOL " +
    token.sval);

```

```

    }
    if
    (token.sval.equals("ACCELERATOR"))
    {
5       System.out.println("Lese Accelerator");
        readAccel();
        break;
10    }
        default:
    }
    }

15    in.close();
    System.out.flush();
    System.out.println("EINLESEN
    DER DATEI " + configFile + "
    FERTIG!");
20    //und wichtig für die Anzeige:
        setLayer();
        setAttributNames();
    } catch
25 (FileNotFoundException e) {
        System.err.println( configFile + " is not found");
    } catch (IOException e) {
        e.printStackTrace();
30    }
}

    private void readToolbar(String
    lgcPath) {
35    int c = '{';
        gobjekte.clear();
        //System.out.println("Jetzt
        kommt die Toolbar");
        try {
40            while (c != '}') {
                switch
                (c=token.nextToken()){
                    case StreamTokeniz-
50 zer.TT_WORD:
                        if
                        (token.sval.equals("NODE")) {
                            //System.out.println("Lese Kno-
                            ten");
                            readNode(lgcPath);
                            break;
                        }
                        if
55 (token.sval.equals("EDGE")) {
                            //System.out.println("Lese Kan-
                            te");
                            readEdge(lgcPath);
                            break;
60    }
                        default:
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //c=token.nextToken();
    //System.out.println("IN
    der TOOLBAR " + c );
    } catch (IOException e) {
        e.printStackTrace();
    }
70    //System.out.println("Fertig
    Toolbar");
}

75    private void readNode(String
    lgcPath) {
        int c = '{';
        String typename = new
        String();
80    String image = new String();
        Vector ecken = new Vector();
        Vector konnektoren = new Vec-
        tor();
        Vector konnektorNamen = new
85 Vector();
        Attribute attribute = new
        StandardAttribute();
        Color color = new Co-
        lor(255,255,255);
90    //System.out.println("Ein
        Knoten");
        try {
            while (c != '}') {
                switch (c){
                    case StreamTokeniz-
95 zer.TT_WORD:
                        // Wird nicht mehr be-
                        nötigt
                        // if
100 (token.sval.equals("TYPE")) {
                            //
                            c=token.nextToken();
                            // Sy-
                            stem.out.println("Lese TYPE" +
105 token.sval);
                            // break;
                            // }
                            if
                            (token.sval.equals("NAME")) {
110 c=token.nextToken();
                                    typename = new
                                    String(token.sval);
                                    // Sy-
115 stem.out.println("Lese NAME" +
                                    typename);
                                    break;
                                }
                                if
                                (token.sval.equals("ATTRIBUTES"))
                                {
                                    attribute = new
                                    StandardAttribute();
120    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

21

```

attribute);
        kno-
ten.setColor(color);
5      // Sy-
stem.out.println("Setze Farbe " +
color);
        // Erzeuge Button
mit Werkzeug für Werkzeugleiste
10     // Der Button greift
über den typnamen auf den richti-
gen
        // Knoten zu.
        ToolButton b = new
15 ToolButton(lgcPath + "images/" +
image,

typname,

20 new KnotenTool(editor, typname),

editor.getToolBar());
        edi-
tor.getToolBar().addToolButton(b)
25 ;
        // Eintrag in die
Hashtabelle
        gobjek-
te.put(typname, knoten);
30 //System.out.println("In Hashta-
belle: " + gobjekte);

        break;
35     }
        if
(token.sval.equals("FILLEDOVAL"))
{
        int breite=10;
40     int hoehe=10;
        while
((c=token.nextToken()) != ' ') {
        breite =
45 (int)token.nval;
c=token.nextToken();
        hoehe =
(int)token.nval;
50 stem.out.println("Lese OVAL_FILL"
+ token.nval);
        }
        // jetzt sollten
alle Daten da sein, und es
55     // kann ein Knoten-
prototyp erzeugt werden.
        GraphObjekt knoten
= new FilledOvalKnoten(typname,
60 hoehe,

breite,

```

```

konnektoren,
65 konnektorNamen,

attribute);
        kno-
70 ten.setColor(color);
        // Sy-
stem.out.println("Setze Farbe " +
color);
        // Erzeuge Button
mit Werkzeug für Werkzeugleiste
75 ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

80 typname,

new KnotenTool(editor, typname),

editor.getToolBar());
85     edi-
tor.getToolBar().addToolButton(b)
;
        // Eintrag in die
Hashtabelle
90     gobjek-
te.put(typname, knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);
95     break;
        }
        if
(token.sval.equals("OVAL")) {
100     int breite=10;
        int hoehe=10;
        while
((c=token.nextToken()) != ' ') {
        breite =
105 (int)token.nval;
c=token.nextToken();
        hoehe =
(int)token.nval;
110     // Sy-
stem.out.println("Lese OVAL" +
token.nval);
        }
        // jetzt sollten
115 alle Daten da sein, und es
        // kann ein Knoten-
prototyp erzeugt werden.
        GraphObjekt knoten
= new OvalKnoten( typname,
120 hoehe,

breite,

```



```

        Color color = new Color(255,255,255);
        try {
            while (c != ')') {
                switch (c){
                    case StreamTokenizer.TT_WORD:

                        if
10 (token.sval.equals("NAME")) {

                            c=token.nextToken();
                            typename = new
                                String(token.sval);
15 // Sy-
                                stem.out.println("Lese NAME" +
                                    typename);
                                    break;
                                }
                                if
20 (token.sval.equals("ATTRIBUTES"))
                                {
                                    attribute = new
                                StandardAttribute();
25 while
                                ((c=token.nextToken()) != ')') {
                                    String aname =
                                new String(token.sval);
                                    c = to-
30 ken.nextToken();
                                    String wert = new
                                String(token.sval);
                                    attribu-
                                te.addAttribute(aname,wert,true);
35 attributeName.put(aname,an-
                                ame);
                                    // Sy-
                                stem.out.println("Lese Attribute"
                                + attribute);
40 }

                                    break;
                                }
                                if
45 (token.sval.equals("IMAGE")) {

                                    c=token.nextToken();
                                    image = new
                                String(token.sval);
50 // Sy-
                                    stem.out.println("Lese IMAGE" +
                                        image);
                                        break;
                                    }
                                    if
55 (token.sval.equals("ARROW")) {
                                        int radius = 10;
                                        int winkel = 10;
                                        while
60 ((c=token.nextToken()) != ')') {
                                            radius =
                                        (int) token.nval;

```

23

```

65      c=token.nextToken();          winkel =
      (int)token.nval;
      // Sy-
stem.out.println("Lese Arrow" +
radius+ winkel);
70      }
      // jetzt sollten
alle Daten da sein, und es
      // kann ein Kanten-
prototyp erzeugt werden.
75      GraphObjekt kante =
new PfeilKante(typname,

radius,

80 winkel,

attribute);
      kan-
te.setColor(color);
85      // Sy-
stem.out.println("Setze Farbe " +
color);
      // Erzeuge Button
mit Werkzeug für Werkzeugleiste
90      ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typname,

95 new KantenTool(editor,typname),
editor.getToolBar());
      edi-
100 tor.getToolBar().addToolButton(b)
;
      // Eintrag in die
Hashtabelle
      gobjek-
105 te.put(typname,kante);

//System.out.println("In Hashta-
belle: " + gobjekte);

110      break;
      }
      if
(token.sval.equals("POINT")) {
      int durch = 10;
115      while
((c=token.nextToken()) != ' ') {
      durch =
(int)token.nval;
      // Sy-
120 stem.out.println("Lese Point" +
durch);
      }
      // jetzt sollten
alle Daten da sein, und es

```

```

        break;
    }
    default:
    } //switch
5      c=token.nextToken();
    // Sy-
    stem.out.println("NAECHSTES
    TOKEN" + token.sval);
    } //while
10     //c=token.nextToken();
    } catch (IOException e) {
        e.printStackTrace();
    }
    // System.out.println("Bende
15 readEdge");

    } //readEdge

20 private void readMenu() {
    tools.clear();
    int c = '(';
    try {
        while
25 ((c=token.nextToken()) != ')') {
        //c=token.nextToken();
        String namen = to-
        ken.sval;
        System.out.println("Jetzt
30 kommt das Menu"+ namen);
        c = token.nextToken();
        String aufruf = to-
        ken.sval;
        System.out.println("Jetzt
35 kommt das Menu"+ aufruf);
        tools.put(new
        String(namen), new
        String(aufruf));
    }
40 } catch (IOException e) {
    e.printStackTrace();
}

45 }

    private void readAnalyse() {
        System.out.println("Jetzt
        kommt die Analyse");
    }
50 }

    private void readShorts() {
        System.out.println("Jetzt
        kommt die Shortcut");
    }
55 }

    private void readAccel() {
        System.out.println("Jetzt
        kommen die Accelerators");
    }
60 }

```

```

25 // private void uebergebe
    (String mpunkt,String
    icon1,String icon2) {
    // public void addBut-
65 ton(String menuePunkt, String
    imagel, String image2)

    private void uebergebe(String
    auswahl,String name,String style,
70 int size) {
    int styleInt = 0;
    switch (style.charAt(0)){
        case 'B':
            styleInt = Font.BOLD;
            break;
75 case 'P':
            styleInt = Font.PLAIN;
            break;
        case 'I':
            styleInt = Font.ITALIC;
            break;
            default:
                styleInt = Font.PLAIN;
    }
80 Font font = new Font(name,
    styleInt, size);
    switch (auswahl.charAt(0)){
        case 'M':
            edi-
90 tor.getMenueleiste().setFont(font
    );
            break;
        case 'P':
            //noch zu Implementiern
            break;
95 case 'S':
            edi-
            tor.getStatusleiste().setFont(fon
            t);
            break;
    }
100 }

    private void uebergebe(String
105 auswahl,int r,int g,int b) {
        if (auswahl.equals("PAPER")){
            edi-
            tor.getZeichenflaeche().setBackgr
            ound(new Color(r,g,b));
        }
110 if (auswahl.equals("GRID")){
            //noch zu implementiern
        }
        if
115 (auswahl.equals("MENUBGC")){
            // edi-
            tor.getMenueleiste().setBackgroun
            d(new Color(r,g,b));
        }
120 if
        (auswahl.equals("MENUFGC")){

```

```

    /**
     * Liefert alle anzeigbaren
AttributNamen zurück.
    */
5    public Enumeration getAttributNames() {
        return attributNamen.keys();
    }

10    /**
     * Liefert die maximale Anzahl
der Attribute zurück.
    */
    public int countAttributNamen() {
15        return attributNamen.size();
    }

    /**
20    * Fügt einen Attribut namen
in die

```

## 2. Datei "load"

```

45    package commands;

import etc.*;
import java.util.*;
import java.awt.*;
50    import java.io.*;
import interfaces.*;

    /**
     * Lädt einen Graphen aus einer
55    .lgf Datei.
    */
    public class Load extends Befehl
    {
        Vector undo;

60        public Load(GraphEditor editor) {
            super(editor);
            undo=new Vector();
65            help =
"<filename.lgf/.lgc/.lgt>";
        }

70        public void ausfuehren(String[] param) {
            //System.out.println(param);
            int anzahl = param.length;
            switch (anzahl) {
75                case 0 : // bei keinem Argument tun wir nichts.
                    break;
                case 1 : // bei einem Argument wird erst nachgeschaut!

```

```

27        * Hashtabel ein.
        */
        public void addAttributName (
25        String name) {
            attributNamen.put(new
String(name), new String(name));
        }

30        /**
         *
        */
        public Hashtable getTools() {
            return tools;
35        }

        // public String getConfigFile()
        {
            // return configFile;
40        // }
    }

```

```

80        if
(param[0].endsWith(".lgc") ||
        param[0].endsWith(".lgf") ||
        param[0].endsWith(".lgt") ) {
85            // wir wurden
von der CommandoZeile aufgerufen
            File file = new
File(param[0]);
90            //System.out.println("Der Pfad :
" + file.getParent());

            //System.out.println("Der Name :
95            " + file.getName());

            pruefe(file.getParent()+"/", file.getName());

        } else {
            //nothing
        }
        break;
        default : //zuviel Parameter
            break;
100        } //switch
    }

    public void ausfuehren(String
105    param) {
        editor.getStatusleiste().show("Load.
..");
115        ((Component) editor).setCursor(Cur

```

```

    } // redo

    /**
     * Diese Klasse wird leider
5 nicht an
     * die Windows bzw Solaris Kom-
     * ponente
     * weitergereicht.
     */
10 class lgFilter implements Fi-
    lenameFilter {
        public boolean accept (File
            dir, String name) {
            return ( na-
15 me.endsWith(".lgf") ||
                na-
                me.endsWith(".lgc") ||
                na-
                me.endsWith(".lgt") );
20 }
    }
    /**
     * Diese Methode überprüft, ob
     * die richtige
25 * Konfigurationsdatei geladen
     * ist, ansonsten wird
     * versucht die richtige zu la-
     * den. (->Editor zurücksetzen)
     * Dannach wird die gewünschte
30 .lgt oder .lgf Datei
     * geladen.
     */
    private void pruefe (String
        pfad, String datei) {
35 Einstellungen settings= edi-
        tor.getEinstellungen();
        if (datei.endsWith(".lgc")) {
            //System.out.println("eine
            lgc Datei");
40 File f = new File(pfad +
            datei);
            if (f.exists()) {
                settings.appName = Ein-
                stellungen.format(datei);
45 settings.fileName=" ";
                settings.frameName = set-
                tings.fileName+ " "
                +settings.appName + " "
                +settings.copyright;
50 settings.configFile = new
                String(datei);
                settings.lgcPath = new
                String(pfad);
                //wir Starten den Editor
55 neu
                editor.start();
            } else {
                System.err.println("File
                not found : "+ settings.lgcPath +
60 datei);
            }
        }
    }

```

29

```

    } else if
        (datei.endsWith(".lgf")) {
        //System.out.println("eine
65 lgf Datei");
        File f = new File(pfad +
            datei);
            if (f.exists()) {
                settings.fileName = da-
70 tei;
                // wir holen uns noch den
                namen des .lgc Files:
                String config = edi-
                tor.getDateischnittstelle().getCo-
                nfig(pfad + datei);
75 //System.out.println("Der
                neue Name der Lgc datei " + con-
                fig);
                f = new
80 File(settings.lgcPath + config);
                if (f.exists()) {
                    // ist diese lgc Datei
                    schon geladen?
                    if
85 (settings.configFile.equals(confi-
                    g)) {
                        //wir muessen nur die
                        lgf Datei laden
                        edi-
90 tor.getDateischnittstelle().load(
                        pfad,datei,editor.getGraph());
                        settings.frameName =
                        settings.fileName+ " "
                        +settings.appName + " "
95 +settings.copyright;
                        ((Frame)editor). set-
                        Title(settings.frameName);
                    } else {
                        // wir müssen auch
100 die Konfigurationsdatei laden
                        settings.appName =
                        Einstellungen.format(config);
                        settings.configFile =
                        new String(config);
                        settings.frameName =
105 settings.fileName+ " "
                        +settings.appName + " "
                        +settings.copyright;
                        //wir Starten den
110 Editor neu
                        editor.start();
                        edi-
                        tor.getDateischnittstelle().load(
                        pfad,datei,editor.getGraph());
115 }
                    } else {
                        Sy-
                        stem.err.println("File not found
                        : " + settings.lgcPath + config);
                    }
                } else {
                    System.err.println("File
                    not found : " + pfad + datei);

```

```

    * Die Toolbar ermöglicht das
    hinzufügen und entfernen
    * von ToolButtons, und deren zu-
    gehörigen ActionListener.
5  */
    public class Toolbar extends Panel {
        GraphEditor editor;
        Tool currentTool;
10    ToolButton currentButton;
        int borderSize = 4;
        /**
         * Der Konstruktor erzeugt das
         AuswahlTool,
15    * da dieses immer vorhanden
         sein sollte.
         */
        public Toolbar(GraphEditor editor) {
20    this.editor = editor;
            setLayout(new BorderLayout(BarLayout.
                VERTIKAL, 2));
            setBackground(editor.getEinstellungen().toolBarBgCo);
25    // eine kleine Lücke
            add(new Space(5, 24));

            ToolButton b = new ToolButton(editor.getEinstellungen().lgc
30    Path +

            "images/auswahl.gif",

35    "Select",

            new AuswahlTool(editor), this);
            setCurrentTool(b.getTool());
            setCurrentButton(b);
40    add(b);
            add(new Space(5, 24));
        }

45    public Insets getInsets() {
        Insets insets =
        (Insets) (super.getInsets()).clone
        ();
        insets.top += borderSize;
50    insets.left +=
        (borderSize + 2);
        insets.bottom += borderSize;
        insets.right +=
        (borderSize + 2);
55    return insets;
        }

        public void paint(Graphics g) {
60    super.paint(g);
            Insets insets = super.getInsets();

```

31

```

        int w = getSize().width -
        insets.left - insets.right;
        int h = getSize().height -
65    insets.top - insets.bottom;

        g.setColor(editor.getEinstellungen().toolbarBgCo);
        for (int i = 0; i < borderSize;
70    i++) {

            g.draw3DRect(i + insets.left, i + insets.top,
                                w - 2 * i - 1, h -
75    2 * i - 1, i < borderSize / 2);
        }

        /**
         * Fügt einen ToolButton hinzu.
         */
80    public void addToolButton(ToolButton button) {
        add(button);
85    }

        /**
         * Entfernt einen ToolButton.
         */
90    public void deleteToolButton(ToolButton button) {
        }

        /**
         * Setzt das aktuelle Tool;
         * wird normalerweise von den
         ToolButtons aufgerufen.
         */
100    public void setCurrentTool(Tool
        currentTool) {
        this.currentTool = currentTool;
        this.currentTool.reset();
        }

105    /**
         * Setzt den aktuellen Button,
         damit der nächste
         * aktuelle Button ihn zurück-
         setzen kann.
         */
110    public void setCurrentButton(Tool
        Button currentButton) {
        if (this.currentButton !=
115    null)
            this.currentButton.setUp();
            this.currentButton = currentButton;
            this.currentButton.setDown();
120    }

        /**

```

**Patentansprüche**

1. Verfahren zur Bestimmung einer graphischen Struktur eines technischen Systems,
  - 5 a) bei dem aus einer Menge mehrerer unterschiedlicher Graphen-Struktur-Dateien eine Graphen-Struktur-Datei ausgewählt wird, wobei in einer Graphen-Struktur-Datei jeweils angegeben ist, welche Elemente zu deren Darstellung ausgewählt werden können, um das technische System
  - 10 in seiner Struktur graphisch zu beschreiben,
  - b) bei dem Elemente ausgewählt werden derart, daß mit den ausgewählten Elementen das technische System beschrieben wird, und
  - 15 c) bei dem die Elemente von einem Editor-Programm dargestellt werden, in welches die ausgewählte Graphen-Struktur-Datei eingebunden worden ist, womit die graphische Struktur des technischen Systems bestimmt ist.
2. Verfahren nach Anspruch 1,
  - 20 bei dem das technische System eine elektronische Schaltung ist.
3. Verfahren nach Anspruch 2,
  - bei dem das technische System eine technische Anlage ist.
4. Verfahren nach einem der Ansprüche 1 bis 3,
  - bei dem die Elemente Graphenelemente eines Graphen sind, die das technische System beschreiben.
- 30 5. Verfahren nach einem der Ansprüche 1 bis 4,
  - bei dem die bestimmte graphische Struktur des technischen Systems auf vorgegebene Strukturregeln hin überprüft wird.
6. Anordnung zur Bestimmung einer Graphen-Struktur,
  - 35 a) mit einem Speicher, in dem eine Menge mehrerer unterschiedlicher Graphen-Struktur-Dateien gespeichert sind, wobei in einer Graphen-Struktur-Datei jeweils angegeben

- eine Auswahleinheit, mit der eine Graphen-Struktur-Datei aus der Menge der Graphen-Struktur-Dateien ausgewählt werden kann,
- ein Editor-Programm, mit dem unter Verwendung einer  
5 aus der Menge der Graphen-Struktur-Dateien ausgewählten Graphen-Struktur-Datei ein Graph mit Elementen der ausgewählten Graphen-Struktur-Datei bestimmt werden kann, womit die Graphen-Struktur bestimmt ist,
- eine mit dem Editor-Programm gekoppelte Darstellungs-  
10 komponente, mit der die bestimmte Graphen-Struktur dargestellt werden kann.

11. Satz von Anordnungen nach Anspruch 10,  
bei dem die erste Anordnung und die zweite Anordnung über ein  
15 Kommunikationsnetz miteinander gekoppelt sind.

12. Satz von Anordnungen nach Anspruch 10 oder 11,  
bei dem mit dem Graphen eine Struktur eines technischen Systems beschrieben wird.

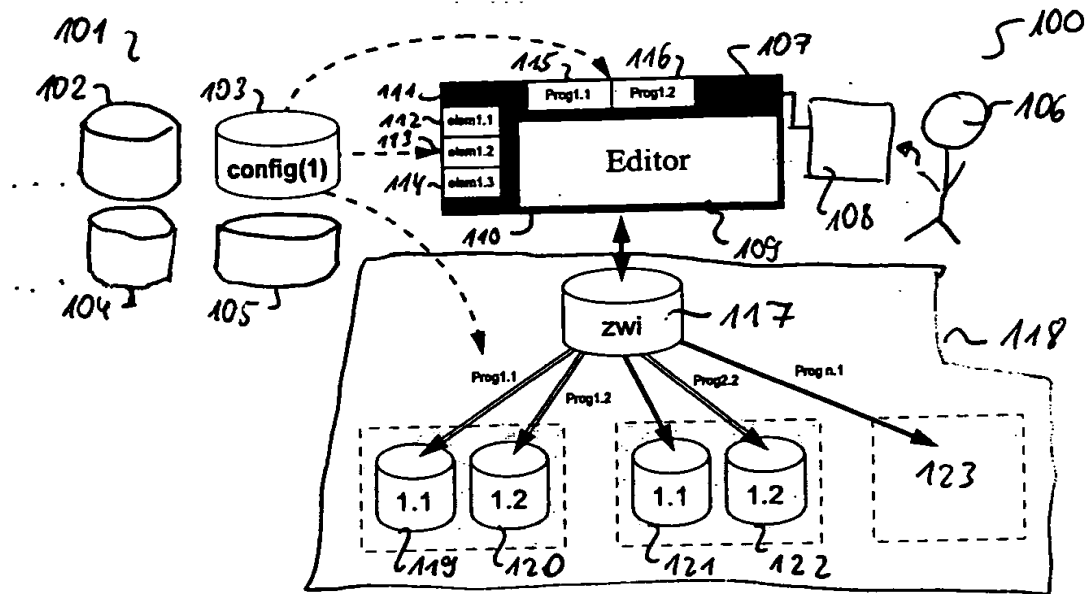
20

13. Anordnung nach Anspruch 12,  
bei dem das technische System eine elektronische Schaltung ist.

14. Anordnung nach Anspruch 12,  
bei dem das technische System eine technische Anlage ist.

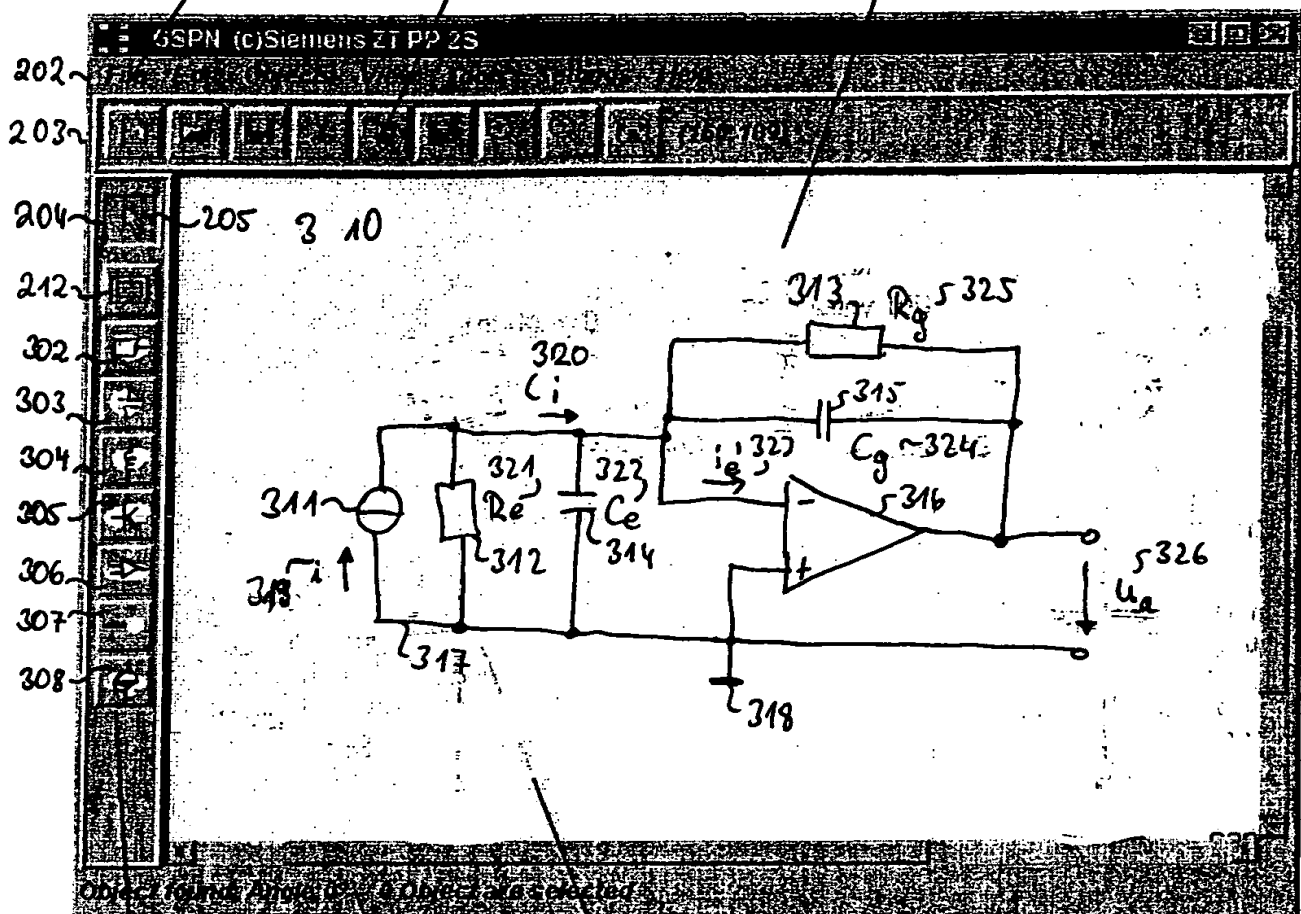
115

FIG 1





3/5



5/5

FIG 5

